Zero-Knowledge Blind Identification For Smart Cards Using Bilinear Pairings

Amitabh Saxena and Ben Soh
Computer Science and Computer Engineering
La Trobe University
Bundoora, VIC, Australia 3086

Serguey Priymak
Applied Science Department
RMIT University
Melbourne, VIC, Australia 3000

December 22, 2005

Abstract

In identification protocols with public verifier coins (like Fiat-Shamir), a passive adversary watching the communication gains information intended only for the verifier. On the other hand, private coin protocols with fewer than three rounds cannot be zero-knowledge. In this paper, we introduce the notion of *bounded-prover* zero-knowledge proofs which require only two rounds and can be considered perfectly zero-knowledge under certain intractibility assumptions. Specifically, we exploit the gap between two computational problems to achieve zero-knowledge in a dishonest verifier scenario. Our example is based on the apparant intractibility of the Linear Diffie-Hellman Problem in bilinear maps. As a natural extension of the single user identification, we present the concept of 'all or none' group identification protocol that can be used to authenticate together an arbitrary number of users in a batch. We also present some extensions of our scheme.

1 Introduction

Zero-knowledge proofs [1, 2, 3] have both theoretical and practical implications. For instance, in a cryptographic protocol, zero-knowledge provides us some assurance of the robustness of the protocol. We observe, however, that either the existing notions of zero-knowledge proofs do not sufficiently capture the security requirements of many protocols or that most protocols do not sufficiently adhere to the theoretical definition of zero-knowledge proofs. We see this as a limitation of our definitions of zero-knowledge. We elaborate this below.

Typically, zero-knowledge proofs assume that the prover is computationally unbounded and the verifier is only allowed to toss public coins. However, in many practical scenarios, it is necessary for the verifier to toss secret coins. To prove security (or essentially, zero-knowledge) of cryptographic protocols in this scenario, new intractability assumptions need to be used; for instance the existence of a "secure" encryption function or the use of random oracles. Likewise, using 'compilers' [4] by increasing the number of rounds to convert an honest verifier zero-knowledge protocol to dishonest-verifier zero-knowledge protocol is inefficient in practice.

Our Contribution

In this paper, we study zero-knowledge proofs with bounded provers and secret (verifier) coin-tosses and propose an identification protocol using bilinear pairings based on this study. Our proposed protocol has several advantages:

1. Our protocol is provably perfect zero-knowledge in our new notion of bounded-prover zero-knowledge [5] under the LDHP hypothesis (see section 4).

2. It is a two-round identification protocol (the verifier issues a challenge and the prover replies). To date, all perfect zero-knowledge identification protocols (including Fiat-Shamir [6]) are three-round (prover sends initial message, verifier issues challenge, prover replies). Contrary to popular belief that two-round perfect zero-knowledge protocols are not possible for non-trivial languages, we show that two-round protocols can be classified perfectly zero-knowledge under some reasonable assumptions.

3. The protocol is "blind" to a passive adversary; no passive adversary watching the communication gains any "useful" information on the outcome of the identification. That is, it is impossible for the adversary to tell if the verifier is honest or not. Similarly it is also impossible to tell if the prover is honest or not.[1] We note that neither the Fiat-Shamir protocol nor any of the other (perfect zero-knowledge) identification protocols provide this blindness property.

4. Our protocol is based on exactly the same infrastructure as the BLS scheme [7]. Thus, the two schemes can be used in conjunction.

The rest of the paper is organized as follows. In section 2 we give some notations and concepts necessary to describe our protocol. In section 4 we present the cryptographic primitives used in our protocol along with the necessary hardness assumptions. We describe the underlying Public Key Infrastructure required for our protocol in section 4.1. Finally, we present our scheme in section 4.2 and provide several extensions in sections 6 and 7.

2 Background

In this section, we present a simple two-round identification scheme using a public key cryptosystem. Assume that Alice and Bob are two users and Alice wishes to identify herself to Bob. We only consider one-way identification and ignore the case of Bob identifying himself to Alice.

First we give some notation. If $\mathbb{A}$ is a non-empty set, then $x \leftarrow \mathbb{A}$ denotes that x has been uniformly chosen in $\mathbb{A}$. A *round* of a protocol involves the exchange of one message. A sequence of two synchronous (ordered) message transmissions constitutes two separate rounds while any number of asynchronous messages (i.e. messages that can be unordered) are part of the same round. A single message passing is a one-round protocol.

2.1 Two-Round Identification With Secret Coins

Alice has public encryption function E_a and a secret decryption function D_a. Let $\mathcal{H} : \{0,1\}^* \mapsto \{0,1\}^*$ be a cryptographic hash function.

1. Alice begins by claiming to know D_a.

2. Bob tosses secret coins to generate a random challenge message m and encrypts it using E_a to get ciphertext $c = E_a(m)$. Bob sends c to Alice over an insecure channel.

3. Alice generates a random $r \leftarrow \{0,1\}^*$ and computes $s = \mathcal{H}(D_a(c)\|r)$. She sends back the tuple $\langle r, s\rangle$ to Bob over the same insecure channel.

4. Bob accepts if $s = \mathcal{H}(m\|r)$.

Observe that Alice must toss coins to insert randomness r into her reply to avoid the known ciphertext attack. The identification protocol presented in this paper is based on a similar idea. However, instead of the difficulty of inverting the public encryption function, we rely on the difficulty of computing discrete logarithms in certain groups. Like the above protocol, our scheme is two-round and begins by the verifier tossing secret coins while the prover then tosses public coins.

[1] Of course, the prover should be able to decide if the verifier is honest or not. Our protocol allows this. Additionally, if the verifier is dishonest and the prover does not respond, then it gives some information to the adversary. The protocol is blind assuming a prover replies irrespective of whether the verifier is honest or not. In case the verifier is dishonest, the prover simply replies with randomly selected string.

3 Bounded Prover Zero-Knowledge

In this section, we will formalize the notion of Bounded Prover Zero-Knowledge (BP-ZK). Although, the name is misleading, since by "bounded-prover" we implicitly imply "bounded-prover, secret (verifier) coin toss", we drop the latter for notational convenience.

Notation

1. We say a real valued function $f : \mathbb{Z} \mapsto [0,1)$ is *negligible* if there is no polynomial p such that $\forall x \in \mathbb{Z}, f(x) \geq |1/p(x)|$.

2. If $\mathbb{A}$ is a non-empty set, then $x \stackrel{R}{\leftarrow} \mathbb{A}$ denotes that x has been uniformly chosen in $\mathbb{A}$. Fix the alphabet $\Sigma = \{0,1\}$. The set of strings over Σ is denoted by Σ^*. For any string $s \in \Sigma^*$, the symbol $|s|$ denotes the bit-length of s.

3. Let $L \in NP$ be some language. For any $x \in L$, we denote by $\mathcal{W}(L,x)$, the witness that $x \in L$. We denote by **Sample**, a randomized PPT algorithm that outputs a pair (w,x) such that $w = \mathcal{W}(L,x)$. Note that there may be more than one witness to x.

Let $\mathcal{P}$ and $\mathcal{V}$ be two PPT machines (intuitively, the 'prover' and 'verifier'). Define the following game between $\mathcal{P}$ and $\mathcal{V}$. Assume that the setup phase is performed by a trusted third party.

1. *Setup phase*: Generate $(w,x) \stackrel{R}{\leftarrow}$ **Sample**. The prover $\mathcal{P}$ is given (w,x) and x is made public.

2. *Proof phase*: The following 2-round protocol is *sequentially* repeated arbitrary number of times between the prover and random verifiers. Each run i of the protocol between prover $\mathcal{P}$ and some verifier $\mathcal{V}^*$ is elaborated below:

   (a) *Challenge*: $\mathcal{V}^*$ tosses secret coins s_i to compute challenge c_i using a deterministic *challenge generator* **Challenge**. Formally, $s_i \stackrel{R}{\leftarrow} \Sigma^*; c_i \leftarrow \mathbf{Challenge}(x, s_i)$.

   (b) *Response*: On receiving c_i, the prover $\mathcal{P}$ uses a PPT algorithm $\mathbf{Verify}_{\mathcal{P}} : \Sigma^* \times \Sigma^* \times \Sigma^* \mapsto \{0,1\}$ to "verify" the challenge and responds as follows.

      i. If $\mathbf{Verify}_{\mathcal{P}}(w,x,c_i) = 1$, it computes a proof π_i using the triplet (w,x,c_i) and a randomized *proof generator* **Proof**. Formally, $\pi_i \stackrel{R}{\leftarrow} \mathbf{Proof}(w,x,c_i)$

      ii. If $\mathbf{Verify}_{\mathcal{P}}(w,x,c_i) = 0$, it generates $\pi_i \stackrel{R}{\leftarrow} \Sigma^*$.

   $\mathcal{P}$ responds with π_i as its proof.

   (c) *Result*: On receiving π_i, the verifier $\mathcal{V}^*$ uses a PPT algorithm $\mathbf{Verify}_{\mathcal{V}} : \Sigma^* \times \Sigma^* \times \Sigma^* \mapsto \{0,1\}$ to "check" the proof. It accepts if $\mathbf{Verify}_{\mathcal{V}}(x, s_i, \pi_i) = 1$ and rejects otherwise.

Definition 1. *The above protocol is a Bounded-Prover Interactive-Proof (BP-IP) between bounded PPT Turing machines $\mathcal{P}$ and $\mathcal{V}^*$ if the following two conditions hold:*

1. *Completeness*: For all $x \in L$ and $w = \mathcal{W}(L,x)$,

$$Pr\Big[s_i \stackrel{R}{\leftarrow} \Sigma^*; c_i \leftarrow \mathbf{Challenge}(x, s_i); \pi_i \stackrel{R}{\leftarrow} \mathbf{Proof}(w,x,c_i) : \mathbf{Verify}_{\mathcal{V}}(x, s_i, \pi_i) = 1\Big] = 1 \qquad (1)$$

2. *Soundness*: For all x, the following probability is negligible in $|x|$.

$$Pr\Big[s_i, x \stackrel{R}{\leftarrow} \Sigma^*; c_i \leftarrow \mathbf{Challenge}(x, s_i); \pi_i \leftarrow \mathcal{P}(x, c_i) : \mathbf{Verify}_{\mathcal{V}}(x, s_i, \pi_i) = 1 \wedge x \notin L\Big] \qquad (2)$$

In other words, the probability of $\mathcal{V}^*$ accepting proof π_i as valid is negligible unless $\mathcal{P}$ "knows" w. We will now attempt to define zero-knowledge in this model. First we define the probabilities:

$$\epsilon_v = Pr\Big[x \xleftarrow{R} L; c_i \leftarrow \mathcal{V}^*(x) : \mathbf{Verify}_{\mathcal{P}}(\mathcal{W}(L,x), x, c_i) = 1\Big]$$
$$\epsilon_p = Pr\Big[s_i, x \xleftarrow{R} \Sigma^*; c_i \leftarrow \mathbf{Challenge}(x, s_i); \pi_i \leftarrow \mathcal{P}(x, c_i) : \mathbf{Verify}_{\mathcal{V}}(x, s_i, \pi_i) = 1 \wedge x \notin L\Big]$$

Clearly, ϵ_v is the probability that a dishonest verifier can convince an honest prover to accept the challenge as valid while ϵ_p is the probability that a dishonest prover can convince an *honest* verifier that $x \in L$ (without itself knowing whether $x \in L$ or not). Soundness requires that ϵ_p be negligible.

Definition 2. *The above protocol is Bounded-Prover Perfect Zero-Knowledge (BP-pZK) if and only if (a) it is a BP-IP and (b) the following two conditions are satisfied:*

1. *Honest Verifier Zero-Knowledge*: There exists a PPT simulator **Sim** such that for all $x \in L$,
   (a) $Pr\Big[s_i \xleftarrow{R} \Sigma^*; c_i \leftarrow \mathbf{Challenge}(x, s_i); \pi'_i \xleftarrow{R} \mathbf{Sim}(x, s_i, c_i) : \mathbf{Verify}_{\mathcal{V}}(x, s_i, \pi'_i) = 1\Big] = 1$, and;
   (b) The distributions $\mathbf{Sim}(x, s_i, c_i)$ and $\mathbf{Proof}(\mathcal{W}(L,x), x, c_i)$ are *identical.*
2. *Dishonest Verifier Zero-Knowledge*: For all x, the ratio $\delta = \epsilon_v/\epsilon_p$ is negligible in $|x|$.

For completeness, we also define Bounded-Prover Computational Zero-Knowledge (BP-cZK) to imply proofs where an honest verifier simulator exists and the ratio ϵ_v/ϵ_p (< 1) is non-negligible. On the other hand, the protocol is 'knowledge-revealing' if this ratio is ≥ 1. We will prove that our protocol is perfect zero-knowledge in this notation. We emphasize that the trivial looking statement of condition 2 in the definition of BP-pZK, could in fact, be considered one of the *main* results of this paper. This definition essentially provides us with a method to prove (or disprove) that any two-round protocol is perfectly zero-knowledge irrespective of whether a dishonest verifier simulator exists or not.

4 Bilinear Pairings

Pairing based cryptography is based on the existence of efficiently computable non-degenerate bilinear maps (or 'pairings') which can be abstractly described as follows: Let $\mathbb{G}_1$ be a cyclic additive group of prime order q and $\mathbb{G}_2$ be a cyclic multiplicative group of the same order. Assume that computing the discrete logarithm in both $\mathbb{G}_1$ and $\mathbb{G}_2$ is hard. A bilinear pairing is a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ that satisfies the following properties [7, 8]:

1. *Bilinearity*: $e(aP, bQ) = e(P, Q)^{ab}\ \forall P, Q \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$
2. *Non-degeneracy*: $P \neq 0 \Rightarrow e(P, P) \neq 1$
3. *Computability*: e is efficiently computable

The above properties also imply:

$$e(P + Q, R) = e(P, R) \cdot e(Q, R)\ \forall P, Q, R \in \mathbb{G}_1$$
$$e(P, Q + R) = e(P, Q) \cdot e(P, R)\ \forall P, Q, R \in \mathbb{G}_1$$

Additionally, we assume that it is easy to *sample* elements from $\mathbb{G}_1$. Typically, the map e will be derived from either the modified Weil pairing or the Tate pairing on an elliptic curve over a finite field [7, 8, 9]. Without going into the details of generating suitable curves (since the same parameters of [7] will suffice[2]), we assume that $q \approx 2^{171}$ so that the fastest algorithms for computing discrete logarithms in $\mathbb{G}_1$ take $\approx 2^{85}$ iterations. For the rest of this discussion, we fix $P \neq 0$ as any uniformly chosen generator of $\mathbb{G}_1$. Define the following problems in $\mathbb{G}_1$.

[2]Boneh et al.'s short signature scheme of [7] uses a bilinear map $\mathbb{G}_0 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ such that an efficiently computable isomorphism $\psi : \mathbb{G}_1 \mapsto \mathbb{G}_0$ exists. Their construction can be directly adapted to our's by setting $\mathbb{G}_0 = \mathbb{G}_1$.

1. ***Diffie-Hellman Problem (DHP)***: Given $P, xP, rxP \in \mathbb{G}_1$ for unknowns $x, r \in \mathbb{Z}_q^*$, compute $rP \in \mathbb{G}_1$.[3]

2. ***Decisional Diffie-Hellman Problem (DDHP)***: Given $P, xP, rxP, V \in \mathbb{G}_1$ for unknowns $x, r \in \mathbb{Z}_q^*$, decide if $V = rP$.

3. ***Extended Diffie-Hellman Problem (EDHP)***: Given $P, xP, rxP \in \mathbb{G}_1$ for unknowns $x, r \in \mathbb{Z}_q^*$, compute $r^2P \in \mathbb{G}_1$.

4. ***Extended Decisional Diffie-Hellman Problem (EDDHP)***: Given $P, xP, rxP, U \in \mathbb{G}_1$ for unknowns $x, r \in \mathbb{Z}_q^*$, decide if $U = r^2P$ with probability $> \frac{1}{2}$.

5. ***Linear Diffie-Hellman Problem (LDHP)***: Given $P, xP, rxP \in \mathbb{G}_1$ for unknowns $x, r \in \mathbb{Z}_q^*$, compute any pair $\langle yP, (r+xy)P\rangle$ for some $y \in \mathbb{Z}_q$.

6. ***Linear Decisional Diffie-Hellman Problem (LDDHP)***: Given $P, xP, yP, rxP, Z \in \mathbb{G}_1$ for unknowns $x, y, r \in \mathbb{Z}_q^*$, decide if $Z = (r+xy)P$ with probability $> \frac{1}{2}$.

Figure 1 shows the observed relationship between the different problems. We justify the relationships using theorems 1-3 below. First we note an important observation.

Observation. *The DDHP in easy.*

Proof: It is a well known fact that the DDHP is easy in bilinear maps. Given P, xP, rxP, V, deciding if $V = rP$ is equivalent to deciding if $e(P, rxP) = e(xP, V)$ which can be done in polynomial time.

Theorem 1. *DHP $\iff$ EDHP. In other words, the EDHP is hard if and only if the DHP is hard.*

Proof: First we prove that EDHP $\Rightarrow$ DHP. let $\mathrm{DHP}_P(xP, rxP)$ be the output of an oracle that solves the DHP for some fixed P and arbitrary xP, rxP. We can construct an oracle $\mathrm{EDHP}_P(xP, rxP)$ that solves the EDHP for any tuple $\langle P, xP, rxP\rangle$ as follows:

$$\mathrm{EDHP}_P(xP, rxP) = \mathrm{DHP}_P(\mathrm{DHP}_P(\mathrm{DHP}_P(xP, rxP), xP), rxP).$$

For the converse, let $\mathrm{EDHP}_P(xP, rxP)$ be the output of an oracle that solves the EDHP for some fixed P and arbitrary xP, rxP. We construct a $\mathrm{DHP}_P(xP, rxP)$ oracle that solves the DHP for the tuple $\langle P, xP, rxP\rangle$ as follows:

$$\mathrm{DHP}_P(xP, rxP) = \frac{1}{2}(\mathrm{EDHP}_P(xP, xP + rxP) - \mathrm{EDHP}_P(xP, rxP) - P)$$

.

Theorem 2. *LDHP $\Rightarrow$ DHP. In other words, the LDHP cannot be hard if the DHP is easy.*

Proof: It is trivial to prove that LDHP $\Rightarrow$ DHP by constructing an $\mathrm{LDHP}_P(xP, rxP)$ oracle using a $\mathrm{DHP}_P(xP, rxP)$ oracle as follows: Generate $y \leftarrow \mathbb{Z}_q$. Then,

$$\mathrm{LDHP}_P(xP, rxP) = \langle yP, \mathrm{DHP}_P(xP, rxP) + y(xP)\rangle$$

.

Theorem 3. *LDDHP $\Rightarrow$ DHP and EDDHP $\Rightarrow$ DHP.*

[3]The reader may notice that our statement of the DHP is different from the usually accepted one, which asks finding rxP given (P, xP, rP). It is easy to see that the two definitions are equivalent if the the order of $\mathbb{G}_1$ is prime; denote the original variant of the problem by ODHP and our variant by DHP. Clearly, $\mathrm{ODHP}(P, xP, rP) = \mathrm{DHP}(xP, P, rP)$. Swapping the generators is allowed because the ODHP is random self-reducible in a prime order subgroup.

Proof: First we prove the latter; simply use the DHP oracle to solve the EDHP and decide the EDDHP instance (since there is a unique solution). To prove the former; given P, xP, rxP, yP, Z, use the DHP oracle to output rP from (P, xP, rxP) and decide if $e(Z, P) = e(rP, P) \cdot e(xP, yP)$.

The security of our protocol relies on an unproven hypothesis which we now describe. Define the following statements. The 'R' indicates a positive reduction, while the 'L' denotes a non-reduction.

R1 : DHP $\Rightarrow$ LDHP

R2 : LDDHP $\Rightarrow$ LDHP

L1 : DHP $\not\Rightarrow$ LDHP

L2 : LDDHP $\not\Rightarrow$ LDHP

Clearly from theorem 3, **R1** $\Rightarrow$ **R2** (and so **L2** $\Rightarrow$ **L1**). Here the arrow denotes an 'implies' relationship. An open question at this stage is if the converse is true; that is, does **R2** $\Rightarrow$ **R1**? (i.e. does **L1** $\Rightarrow$ **L2**?) Next, we give our hypothesis in the form of the following conjuncture.

Conjuncure 4. *(LDHP Hypothesis) LDDHP $\Rightarrow$ LDHP if and only if the LDHP is easy.*

If this conjuncture turns out to be true, this would clearly imply that DHP $\Rightarrow$ LDHP if and only if the DHP is easy (since, by theorem 3, we have LDDHP $\Rightarrow$ DHP and if DHP $\Rightarrow$ LDHP, this would also imply LDDHP $\Rightarrow$ LDHP).

To disprove this conjuncture, it would seem natural to try and construct a DHP oracle using an LDHP oracle and show that DHP $\Rightarrow$ LDHP even if the DHP is hard. It appears, however, that constructing a DHP oracle using just an LDHP oracle is infeasible if we are unable to verify the LDHP oracle's outputs (assuming that the oracle outputs different values each time when given the same inputs). Another approach would be to try and exhibit an example where the LDHP is easy but the LDDHP is hard or an example where the LDHP is hard but LDDHP $\Rightarrow$ LDHP. Unfortunately, we have been unable to prove this conjuncture at this stage.

Corollary 4. *Assuming conjuncture 4, EDHP $\not\Rightarrow$ LDHP unless both the problems are easy.*

$$\begin{tikzcd}
DHP\ (Hard) \arrow[r, Leftrightarrow] & EDHP & EDDHP\ (Hard\ ?) \arrow[l, Rightarrow] \\
LDHP\ (Hard\ ?) \arrow[ur, Rightarrow] & & LDDHP\ (Hard\ ?) \arrow[ul, Rightarrow]
\end{tikzcd}$$

Figure 1: Problem Hierarchy

We exploit this apparent "gap" in the two problems (LDHP, EDHP) to construct perfect two-round interactive zero-knowledge proofs as shown in the next section. Before describing our protocol, we make the following three assumptions:

1. **LDHP Assumption**: The LDHP is intractable in $\mathbb{G}_1$
2. **LDDHP Assumption**: The LDDHP is intractable in $\mathbb{G}_1$
3. **EDDHP Assumption**: The EDDHP is intractable in $\mathbb{G}_1$

4.1 Setup PKI

1. The TA selects a security parameter l and uses the BDH parameter generator of [7], which we will call **Params**, to set the system parameters as follows. It generates $\{e, q, \mathbb{G}_1, \mathbb{G}_2\} \leftarrow \mathbf{Params}(1^l)$ where $\mathbb{G}_1, \mathbb{G}_2$ are group descriptions for two groups each of prime order $q > 2^l$ and $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ is a bilinear mapping as defined in section 4. The TA then generates $P \leftarrow \mathbb{G}_1$. If $P \neq 0$ then P is a generator of $\mathbb{G}_1$. The system parameters are $\langle e, q, l, \mathbb{G}_1, \mathbb{G}_2, P\rangle$.

2. Each participant $\mathcal{ID}_i$ generates $x_i \leftarrow \mathbb{Z}_q$ as the private key. The corresponding public key is $Y_i = x_i P \in \mathbb{G}_1$. Each user also obtains a certificate from the CA linking the identity $\mathcal{ID}_i$ and the public key Y_i, for example, using the identification protocol given below.

4.2 Two-Round (Blind) Identification

We are now in a position to describe our identification protocol. Assume that user $\mathcal{ID}$ having secret key $x \in \mathbb{Z}_q$ and public key $Y = xP \in \mathbb{G}_1$ wants to prove to server $\mathcal{S}$, the knowledge of x. Additionally, $\mathcal{ID}$ wants to ensure that no one except the verifier $\mathcal{S}$ gets convinced of this fact from watching the communication. Here $(\mathcal{ID}, \mathcal{S})$ can be considered as a pair of (prover, verifier). The protocol is graphically described in figure 2.

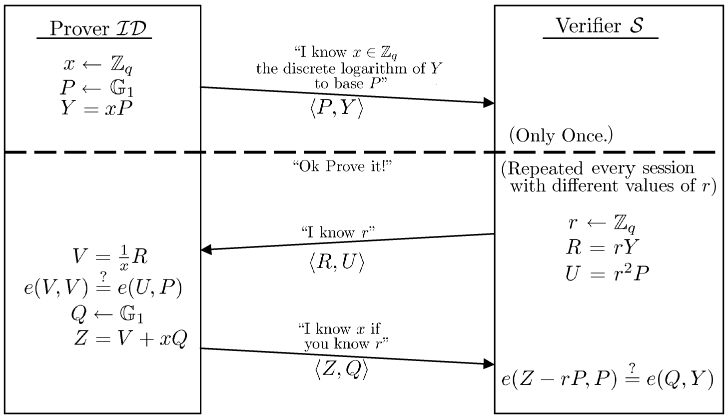

Figure 2: Two-round Identification

1. $\mathcal{ID}$ starts by claiming to know $x \in \mathbb{Z}_q$, the discrete logarithm of $Y \in \mathbb{G}_1$ to base P.

2. The verifier $\mathcal{S}$ generates $r \leftarrow \mathbb{Z}_q$ and computes $R = rY$ and $U = r^2P$. It sends $\langle R, U\rangle$ as its challenge to $\mathcal{ID}$.

3. $\mathcal{ID}$ computes $V = \frac{1}{x}R$ and verifies $e(V, V) = e(U, P)$. If this test passes, $\mathcal{ID}$ is convinced that R was indeed randomly generated. $\mathcal{ID}$ generates $Q \leftarrow \mathbb{G}_1$ and computes $Z = V + xQ$. It sends $\langle Z, Q\rangle$ as its proof to $\mathcal{S}$.

4. $\mathcal{S}$ accepts if $e(Z - rP, P) = e(Q, Y)$.

Correctness: The correctness of the protocol is easily checked. In the verification process;

$$\text{LHS} = e(Z - rP, P) = e(\frac{1}{x}xrP + xQ - rP, P) = e(Q, xP) = \text{RHS}$$

5 Security of the Proposed Protocol

In this section, we prove that the above protocol is perfect zero-knowledge using a restricted definition of Bounded-Prover perfect Zero-Knowledge (BP-pZK) [5], which essentially requires that the probability of a dishonest verifier succeeding is negligibly less then that of a dishonest prover succeeding. The completeness is guaranteed by the correctness of the verification process.

5.1 Soundness

Assuming an honest verifier, we must show that a dishonest prover cannot succeed except with a negligible probability. Given xP, rxP, r^2P, the task of a dishonest prover is to compute a pair $\langle Z', Q' \rangle$ such that $Z' = rP + xQ'$. It is easy to see that this is an instance of the LDHP (section 4). We also claim that knowledge of U does not give a dishonest prover any additional advantage in solving this LDHP instance because deciding if $U = r^2P$ is an instance of the EDDHP. An algorithm that has a non-negligible probability in solving the LDHP given a true EDDHP instance is essentially a distinguisher for the EDDHP instance. Thus, the proof is sound from a verifier's view as long as both the LDHP and the EDDHP are intractable.

5.2 Honest Verifier Zero-Knowledge

We construct a simulator that can generate an accepting transcript $\{P, xP, rxP, r^2P, Z', Q'\}$ without interaction with a prover as follows. Given (P, xP), generate $r, \alpha \leftarrow \mathbb{Z}_q$, compute $rxP, r^2P, Q' = \alpha P$ and $Z' = rP + \alpha xP$. It is easy to see that the simulated and real distributions are identical. Thus, our protocol is *Honest Verifier Zero-Knowledge.*

5.3 Dishonest Verifier Zero-Knowledge

A dishonest verifier will generate R non-uniformly. In other words, a dishonest verifier will not know r corresponding to R. To prove zero-knowledge in this case, it is enough to prove that the probability of a dishonest verifier succeeding is *negligibly* less than the probability of a dishonest prover succeeding.

First, note that it is infeasible for a dishonest verifier to succeed except with a negligible property because computing r^2P from P, xP, rxP (without knowledge of r or x) is exactly an instance of the EDHP.[4]

Let ϵ_v be the probability that a dishonest verifier $\mathcal{S}^*$ succeeds in making an honest prover $\mathcal{ID}$ accept a challenge as valid. Also let ϵ_p be the probability that a dishonest prover $\mathcal{ID}^*$ can convince an honest verifier $\mathcal{S}$. Let Pr[EDHP] and Pr[LDHP] be the probabilities that any computationally bounded adversary can solve the EDHP and the LDHP respectively. From the above discussion, we know that $\epsilon_v = Pr[\text{EDHP}]$ and $\epsilon_p = Pr[\text{LDHP}|(\text{EDDHP instance} = \text{TRUE})] \geq Pr[\text{LDHP}]$

We can straightaway conclude that $Pr(\text{EDHP}) < Pr(\text{LDHP})$ under the LDHP hypothesis using corollary 4 and so the above protocol is computationally zero-knowledge in the BP-cZK model. Now let $\delta = \epsilon_v/\epsilon_p$. Clearly, if δ is non-negligible, it would imply that EDHP $\Rightarrow$ LDHP, again contradicting corollary 4. Hence we conclude that the above protocol is, in fact, perfectly zero-knowledge in the BP-pZK model from a prover's viewpoint.

[4] We observe that the alternate cheating verifier problem; given P, xP, r^2P, computing rxP is at least as hard as the DHP due to the following reduction. Denote this problem by ACVP (for Alternate Cheating Verifier Problem). For simplicity, we will only show how to reduce the Original Diffie-Hellman Problem ($ODHP$) (see footnote in section 4) to the ACVP.

Using an $ACVP(P, xP, r^2P)$ oracle that always outputs rxP for any given inputs P, xP, r^2P, we can easily construct an $ODHP(P, xP, rP)$ oracle that outputs rxP as follows. If r is a quadaratic residue (mod q), then we simply output $ODHP(P, xP, rP) = ACVP(P, ACVP(P, xP, rP), rP)$. If r is a quadratic non-residue (mod q) (in which case the $ACVP$ oracle outputs Error), we generate another quadratic non-residue r' (mod q). Thus, rr' is a quadratic residue (mod q). In this case, we output $ODHP(P, xP, rP) = 1/r' ACVP(P, ACVP(P, xP, rr'P), rr'P)$. Thus, clearly $DHP \Rightarrow ACVP$.

5.4 Passive Adversary Blindness

An inherent property of the above protocol is *passive adversary blindness* which informally implies that no polynomially bounded adversary has a non-negligible advantage in deciding the honesty of the participants in the protocol. Assuming that the EDDHP is intractable, it is impossible for a passive adversary to decide the honesty of the verifier; given P, xP, rxP, U, deciding if $U = r^2P$ is an instance of the EDDHP. Similarly, it is not possible for a passive adversary to decide the honesty of the prover; given P, xP, rxP, Q, Z, deciding if $Z = rP + xQ$ is an instance of the LDDHP.

5.5 Knowledge Extractor

It is trivial to prove that the above interactive protocol is a "proof of knowledge" by constructing an extractor. On a closer look at the protocol, we see that $\mathcal{ID}$ essentially proves knowledge of the witness $\langle rP, xQ\rangle$ such that $\langle Q, Z\rangle \in L_1$ using the shared string $\langle P, xP, rxP\rangle$ where L_1 is the language:

$$L_1 = \{\langle Q, Z\rangle \,|Z = rP + xQ\}.$$

Clearly $L_1 \in NP$. Assume that a dishonest prover ($\mathcal{ID}^*$) is able to make any honest verifier accept. That is, given (P, xP, rxP), $\mathcal{ID}^*$ can always output a pair $\langle Z', Q'\rangle$ such that $e(Z' - rP, P) = e(Q', xP)$. By simulating the honest verifier itself, $\mathcal{ID}^*$ can obtain $\langle rP, xQ'\rangle$, the witness that $\langle Q', Z'\rangle \in L_1$.

6 Two-Round Group Identification

This scheme enables a group of users to identify themselves to a server such that: (a) The identification test passes if none of the users cheat, (b) if any of the users cheat, the test will fail with a high probability, (c) it is not possible for the server or the users to know who cheated.

Assume that $\{\mathcal{ID}_1, \mathcal{ID}_2, \ldots \mathcal{ID}_n\}$ are the set of users who want to jointly identify themselves. It is necessary that each user $\mathcal{ID}_i$ must have a certified public key Y_i as described earlier. The goal of the protocol is that all users will simultaneously identify themselves to a server $\mathcal{S}$. That is, the proof is valid only on all the statements together: "$\mathcal{ID}_i$ knows x_i" $\forall i : 1 \leq i \leq n$ but not on any of the individual statements like "$\mathcal{ID}_1$ knows x_1" or "$\mathcal{ID}_2$ knows x_2" independently of the others. We will assume the infrastructure of section 4.1. The identification is done as follows:

1. The n provers $\mathcal{ID}_1, \mathcal{ID}_2 \ldots \mathcal{ID}_n$ start by claiming to $\mathcal{S}$ that they know the discrete logarithms $x_1, x_2, \ldots x_n \in \mathbb{Z}_q$ of $Y_1, Y_2, \ldots Y_n \in \mathbb{G}_1$ (to base P) respectively.

2. The verifier $\mathcal{S}$ generates $r_1, r_2, \ldots r_n \leftarrow \mathbb{Z}_q$ and computes $R_i = r_iY_i$ and $U_i = r_i^2P$. It makes the list of challenges $\langle \mathcal{ID}_i, R_i, U_i\rangle$ public.

3. Each $\mathcal{ID}_i$ computes $V_i = \frac{1}{x_i}R_i$ and checks that $e(V_i, V_i) = e(U_i, P)$. If this test passes, it generates $Q_i \leftarrow \mathbb{G}_1$ and computes $Z_i = V_i + x_iQ_i$

4. All users then collaborate to jointly compute the value $Z = \sum_{j=1}^{j=n} Z_j$. This computation is hidden from $\mathcal{S}$ so that individual values Z_j are effectively hidden the it's view. The combined proof $\{Z, Q_1, Q_2 \ldots Q_n\}$ is sent to $\mathcal{S}$.

5. $\mathcal{S}$ accepts if $e(Z - \sum_{j=1}^{j=n} r_jP, P) = \prod_{j=1}^{j=n} e(Q_j, Y_j)$.

6.1 Security Proof (Sketch)

We claim that this test will pass if and only if each $\mathcal{ID}_i$ knows x_i. To summarize the goals of the protocol, the individual users can jointly authenticate themselves to the server such that:

(a) If all users are honest, the server always accepts.

(b) If any of the users are dishonest, the server rejects with a high probability.

(c) The protocol is zero knowledge. It is not possible for anyone (including the server) to know which user cheated.

(d) Collusions are possible between users but not with the server (the server is trusted).

The protocol is secure based on the following observations:

1. Soundness: Computing individual proofs $\langle Z_i, Q_i\rangle$ without x_i or r_i is infeasible using similar reasoning for the single user scenario (section 4.2). Using the idea of aggregate signatures of [10], the same applies to the multiuser case. Consequently the verifier will reject.

2. Honest Verifier Zero Knowledge: $\mathcal{S}$ can generate a valid accepting transcript on its own corresponding to $\{r_i, R_i, U_i\}$ $\forall i : 1 \leq i \leq n$ as follows: $\mathcal{S}$ generates $\alpha_i \leftarrow \mathbb{Z}_q$ $\forall i$ and computes $Q_i = \alpha_i P$, $R_i = r_i Y_i$. Then $Z = \sum_{j=1}^{j=n} r_i P + \alpha_i Y_i$.

3. Dishonest Verifier-Zero-knowledge: A dishonest verifier will generate R non-uniformly and will therefore not know r_i corresponding to R_i. Due to this it will be hard for this verifier to generate U_i such that $e(\frac{1}{x_i}R_i, \frac{1}{x_i}R_i) = e(U_i, P)$ due to the hardness of the EDHP. Thus a dishonest verifier will not be able to make anyone accept his/her challenge as valid. The above reasoning for single user identification can be extended here.

4. Honest Verifier Secrecy: We require that it is impossible for a passive adversary to decide the honesty of the verifier. The reasoning for the single user case can be extended here. That is, given P, x_iP, R_i, U_i, deciding if $e(\frac{1}{x_i}R_i, \frac{1}{x_i}R_i) = e(U_i, P)$ is infeasible without knowledge of r_i or x_i due to the hardness of the EDDHP.

5. Honest Prover Secrecy: Assume that all the provers are honest and thus, $\mathcal{S}$ will eventually accept. We require that it is impossible for a passive adversary (including the provers) to decide the honesty some prover. We note that given $P, x_iP, r_ix_iP, r_i^2P, Q_i, Z_i$, deciding if $Z_i = r_iP + x_iQ_i$ is infeasible without knowledge of r or x due to the hardness of the LDDHP. Thus a passive adversary cannot decide the outcome of the identification.

6. Dishonest Prover Secrecy: Assume that some of the provers are dishonest. In this case, given $P, x_iP, r_ix_iP, y_iP, Z_i \in \mathbb{G}_1$, deciding if $Z_i = (r_i + x_iy_i)P$ is infeasible under the LDDHP assumption. Therefore if $\mathcal{S}$ rejects, none of the provers know which pairs $\langle Z_i, Q_i\rangle$ correspond to invalid proofs (if the individual coin tosses r_i of $\mathcal{S}$ are kept secret and $\mathcal{S}$ is honest, no information is leaked to the provers). Similarly if the individual values Z_i are kept secret (from $\mathcal{S}$), the identity of the dishonest provers is still concealed since computing individual proofs Z_i just from $Z, y_1P, y_2P \ldots y_nP$ such that $Z_i = (r_i + x_iy_i)P$ $\forall i$ is infeasible assuming the hardness of the 2-Element Aggregate Extraction Problem (2-EAEP) (which is equivalent to the DHP as shown in theorem 1 of of [11]). Consequently, even the verifier $\mathcal{S}$ does not have the ability to decide which of the provers are dishonest.

   Finally, if the joint computation of Z is carried out in a way that any one individual prover or a small coalition of provers can know Z_i's for only a small fraction of users, the identities of dishonest provers can still be effectively hidden, even if $\mathcal{S}$ can be coerced to reveal all the coin tosses r_i.

7 Other Extensions

In this section we will provide several extensions of our scheme. We refer to the definitions of sections 4.1 and 4.2. The private keys x_i can either be generated by the users or by a trusted authority. The public key Y_i are assumed to be certified in the former case. Note that the identification is a two-round protocol, with the verifier sending the challenge R in the first step and the prover sending the response Z, Q in the second step. The private key for each smart card is encapsulated in a tamper proof chip. Signing access to this key is given via some access control mechanism like a PIN number. The corresponding public key is also present in the smart card along with a certificate. Smart cards may be purchased from a (reputed) third party and must be registered with the relevant authority (like a bank) before they can be used. To register a smart card, the authority simply provides a certificate.

7.1 Hidden Signatures

In the protocol of section 4.2, where user $\mathcal{ID}$ identifies to the server $\mathcal{S}$, $\mathcal{ID}$ can also send plaintext messages along with hidden signatures such that only $\mathcal{S}$ can extract the signature. Of course, once extracted, the signatures provide the same non-repudiation as the BLS signatures [7].

1. ***Initialization***: The process begins when at some point, $\mathcal{S}$ asks $\mathcal{ID}$ to identify itself by sending the challenge $R = rxP$ and $U = r^2P$ in the first step of the protocol of section 4.2.

2. ***Signing***: Like the previous scheme, the message to be signed is $M \in \mathbb{G}_1$ and $Q = \mathcal{H}(M)$. However, in this case, the verifier's challenge is *not* ignored. As always, the signer $\mathcal{ID}$ first computes $V = \frac{1}{x}R$ and checks that $e(V, V) = e(U, P)$. The hidden signature of $\mathcal{ID}$ on M is then $Z = V + xQ$. The tuple $\langle M, Z\rangle$ is sent to $\mathcal{S}$.

3. ***Verification***: On receiving $\langle M, Z\rangle$, $\mathcal{S}$ extracts the signature $S = Z-rP$. The verification condition is $e(S, P) = e(\mathcal{H}(M), Y)$ like before.

7.2 Plaintext-Aware Encryption And Signcryption

The above idea of hidden signatures suggests that we can also easily convert the interactive identification protocol to a non-interactive public key encryption scheme. We present (without a security proof) such a scheme here based on El Gamal encryption. Our variant of El Gamal provides semantic security and additionally achieves *plaintext awareness* (which informally requires that an adversary cannot generate ciphertexts without "knowing" the actual plaintext [12, 13]). The idea is to simulate the identification protocol for an arbitrary user and give the message as input to the simulator. Then construct the challenge for a specific user $\mathcal{ID}$ using the *same* coin tosses of the verifier from the simulation. As usual, we assume user $\mathcal{ID}$ has a private key $x \in \mathbb{Z}_q$ and the corresponding certified public key $Y = xP \in \mathbb{G}_1$ for a known generator P of $\mathbb{G}_1$. Let the plaintext be $M \in \mathbb{G}_1$. We also require a hash function $\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$. The use of the hash function is necessary only to achieve semantic security.

1. ***Encryption***: Generate $r, x' \leftarrow \mathbb{Z}_q$, $P' \leftarrow \mathbb{G}_1$. Compute $Q = \frac{1}{x'}(\mathcal{H}(rP)+M)$, $Z = rP+\mathcal{H}(rP)+M$, $Y' = x'P'$, $R = rY$ and $U = r^2P$. The ciphertext is the tuple $\langle P', Y', R, U, Z, Q\rangle$. The values $\{P', Y'\}$ can be re-used in multiple encryptions to save bandwidth (without any compromise in security). It is also possible to have $P = P'$ which saves further bandwidth. We allow the case of $P \neq P'$ to enable signcryption (described later).

2. ***Decryption***: Compute $V = \frac{1}{x}R$. Check that $e(V, V) = e(U, P)$ and $e(Z - V, P') = e(Q, Y')$. If both checks pass, accept the ciphertext as valid and compute $M = Z - V - \mathcal{H}(V)$.[5]

Plaintext Awareness: An adversary cannot make $\mathcal{ID}$ accept the ciphertext $\langle P', Y', R, U, Z, Q\rangle$ without knowing the corresponding message $M = x'Q - \mathcal{H}(rP)$ such that $Y' = x'P'$. This follows from the various Diffie-Hellman assumptions given earlier.

Semantic Security: Given plaintexts $\{M_0, M_1\}$ and one of the ciphertexts $\langle P', Y', R_b, U_b, Z_b, Q_b\rangle$, $b \in \{0, 1\}$, computing b is equivalent to one of the following:

1. Decide if $M_0 \stackrel{?}{=} Z_b - \frac{1}{x}R_b - \mathcal{H}(\frac{1}{x}R_b)$ with probability greater than 1/2

2. Decide if $\left\langle Q_b, P', M_0 + \mathcal{H}(\frac{1}{x}R_b), Y'\right\rangle$ forms a valid DDH tuple with probability greater than 1/2. We say that $\langle A, B, C, D\rangle$ forms a valid DDH tuple if $e(A, B) = e(C, D)$.

Signcryption: In the above protocol $\langle P', Y'\rangle$ acts as the public key of the user in the simulated identification. If it is a real certified public key then this scheme also serves as a signcryption scheme. The tuple $\langle V, Q, M\rangle$ provides non-repudiation; a verifier can check that $e(M + \mathcal{H}(V), P') = e(Q, Y')$.

[5] In this case $\langle P', Y'\rangle$ acts as the public key of the simulated user. Observe that we have "mixed" the public keys of different users in the identification. This mixing is justified assuming that the order of $\mathbb{G}_1$ is prime. Due to this, all elements of $\mathbb{G}_1$ except 0 will have order q. Alternatively, if $P', Y' \neq 0$ then it is ensured that the hardness assumptions on $\langle P, Y\rangle$ based purely on the order of the cyclic group generated hold equally well for any other pair with the same properties

7.3 Deniable Signcryption

It is possible for some user $\mathcal{ID}_a$ to send an authenticated and encrypted message to $\mathcal{ID}_b$ (with $\mathcal{ID}_b$'s co-operation) such that it can be later denied by $\mathcal{ID}_a$. As before we assume that the private key of $\mathcal{ID}_a$ is x_a corresponding to the public key x_aP_a. Also, let x_b be the private key of $\mathcal{ID}_b$ corresponding to the public key x_bP_b (possibly with $P_a = P_b$).

1. ***Initialization***: At some point $\mathcal{ID}_b$ asks $\mathcal{ID}_a$ to identify itself. To do this it generates $r_b \leftarrow \mathbb{Z}_q$, computes $R_b = r_bY_a$, $U_b = r_b^2P_a$ and sends $\langle R_b, U_b\rangle$ to $\mathcal{ID}_a$.

2. ***Signcryption***: $\mathcal{ID}_a$ computes $V_a = \frac{1}{x_a}R_b$ and accepts the challenge if $e(V_a, V_a) = e(U_b, P_a)$. Using $\mathcal{ID}_b$'s challenge R_b, $\mathcal{ID}_a$ can then encrypt a message $M_a \in \mathbb{G}_1$ for $\mathcal{ID}_b$ as follows: $\mathcal{ID}_a$ generates $r_a \leftarrow \mathbb{Z}_q$, sets $Q_a = M + 2r_aP_b$ and computes $Z_a = V_a + x_aQ_a$ as always. It then computes $T_a = Q_a - r_aP_b$ and $R_a = r_aY_b$ and sends $\langle Z_a, T_a, R_a\rangle$ as its signcrypted message to $\mathcal{ID}_b$

3. ***Verification/Decryption***: On receiving $\langle Z_a, T_a, R_a\rangle$, $\mathcal{ID}_b$ first computes $Q_a = T_a + \frac{1}{x_b}R_a$ and verifies that (Z_a, Q_a) is a valid accepting transcript for $\mathcal{ID}_a$. That is, $e(Z_a - r_bP_a, P_a) = e(Q_a, Y_a)$. If this check passes $\mathcal{ID}_b$ computes $M_a = T_a - \frac{1}{x_b}R_a$. The zero knowledge identification property ensures that $\mathcal{ID}_b$ will only accept messages that were actually sent by $\mathcal{ID}_a$. Thus, an adversary cannot make it accept any random message as valid, thereby ensuring plaintext-awareness.

4. ***Repudiation (Deniablity)***: We will show that the above scheme allows $\mathcal{ID}_a$ to later deny sending the message M_a. Firstly note that $M_a = Q_a - \frac{2}{x_b}R_a$ and $T_a = Q_a - \frac{1}{x_b}R_a$. For any given pair (M_a, Q_a), it is easy to see that $\mathcal{ID}_b$ has the ability to generate $\langle R_a, T_a\rangle$. Due to this, there is no evidence left for $\mathcal{ID}_b$ to prove that these values were actually computed by $\mathcal{ID}_a$.

   Also note that $Z_a - r_bP_a = x_aM_a + 2x_ar_aP_b$. Extracting x_aM_a (which would serve as $\mathcal{ID}_a$'s signature on M_a) using this relation is equivalent to computing $x_ar_aP_b$ from r_aP_b and x_aP_a without knowing r_a or x_a. This is a hard problem of the order of the DHP (see [10], section 4.2 and [11]).

7.4 Two-Round Authenticated Key Agreement

User $\mathcal{ID}_a$ having public key x_aP_a and private key x_a can establish a shared key with user $\mathcal{ID}_b$ having public key x_bP_b and private key x_b as follows. Let $\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$ be a cryptographic hash function.

1. $\mathcal{ID}_a$ generates $r_a \leftarrow \mathbb{Z}_q$ and computes $R_a = r_aY_b$, $U_a = r_a^2P_b$ and $Z_a = r_aP_b + x_a\mathcal{H}(R_a)$. It initiates the protocol by sending $\langle R_a, U_a, Z_a\rangle$ to $\mathcal{ID}_b$. Essentially, $\mathcal{ID}_a$ simulates the identification protocol with itself and sends part of the transcript to $\mathcal{ID}_b$.

2. On receiving $\langle R_a, U_a, Z_a\rangle$ from $\mathcal{ID}_a$, $\mathcal{ID}_b$ computes $V_b = \frac{1}{x_b}R_a$ and $Q_a = \mathcal{H}(R_a)$. It then verifies that $\langle R_a, U_a, Z_a, V_b, Q_a\rangle$ is indeed an accepting transcript of $\mathcal{ID}_a$'s identification. That is, it checks that $e(Z_a - V_b, P_a) = e(Q_a, Y_a)$ and $e(V_b, V_b) = e(U_a, P_b)$. If both tests pass $\mathcal{ID}_b$ accepts $\mathcal{ID}_a$'s authentication. If $\mathcal{ID}_b$ decides to continue with the process it generates $Q_b \leftarrow \mathbb{G}_1$ and $r_b \leftarrow \mathbb{Z}_q$. It then computes $R_b = r_bY_a$, $U_b = r_b^2P_a$ and $Z_b = V_b + r_bP_a + x_b\mathcal{H}(R_b)$. It sends $\langle R_b, U_b, Z_b\rangle$ to $\mathcal{ID}_a$ as its response. It also keeps $K_{ab} = r_bP_a + V_b = r_bP_a + r_aP_b$ as the shared key.

3. On receiving $\langle R_b, U_b, Z_b\rangle$, $\mathcal{ID}_a$ computes $V_a = \frac{1}{x_a}R_b$ and $Q_b = \mathcal{H}(R_b)$ and performs the following two checks: $e(V_a, V_a) = e(U_b, P_a)$ and $e(Z_b - r_aP_b - V_a, P_b) = e(Q_b, Y_b)$. If both checks pass, it accepts $\mathcal{ID}_b$'s authentication and keeps $K_{ab} = r_aP_b + V_a = r_bP_a + r_aP_b$ as the shared key.

We claim that in the second step, $\mathcal{ID}_b$ will accept if and only if $\mathcal{ID}_a$ knows r_a and x_a. To see this, first note that $\langle Z_a, Q_a\rangle$ is a zero knowledge identification proof of $\mathcal{ID}_a$. Due to this, there is no guarantee that the proof was generated by $\mathcal{ID}_a$ (since it could also have been efficiently simulated according to section 4.2). However, observe that if this protocol is simulated, the resulting Q_a will almost certainly be random. A simulator cannot choose a predetermined value of Q_a since there is no way to output an accepting configuration for a specific Q_a without knowledge of x_a under the LDHP assumption. The use of the hash function additionally ensures that the simulator did not have control even over the random coin tosses r_a. Hence, for this particular instance, the simulation must have been carried out by $\mathcal{ID}_a$.

The second and third steps of the protocol involve the identification of $\mathcal{ID}_b$ to $\mathcal{ID}_a$ keeping r_a as the random coin tosses of verifier. The need to include R_b becomes evident when we observe that the first message from $\mathcal{ID}_a$ does not include any session specific information. Thus, authenticating the first message alone cannot guarantee key freshness. We use the technique mentioned in [14], section 3.1 and provide freshness via the computation of the session (or ephemeral) key which includes the 'fresh' value r_bP_a along with the possibly 'stale' value r_aP_b. Due to this, a replay attack is detectable when no message or a garbled message is received by either parties.

7.5 Anonymous Seller Credit Card Payments

In this section, we will present a simple on line payment system with some interesting security features. The protocol requires only one certified key. The seller of a product need not provide a certified key to the buyer, effectively remaining anonymous. The seller must produce some identification to the credit card processor or the bank to ensure that the payment is successful. If the buyer notices a disputed transaction on his credit cart statement, he can ask the bank to reveal the identity of the party who received the money. If the transaction is not disputed, the seller can remain completely anonymous. Moreover, we provide the additional advantage of 'single-use' transactions, that is after having successfully processed a payment, the seller cannot later reuse the same information to process another identical payment. We will assume the identification scheme of section 4.2 where the buyer is $\mathcal{ID}$ and the seller is $\mathcal{S}$. The protocol also involves a third party $\mathcal{B}$ which could be a bank or a credit card processor. The certified public key of $\mathcal{ID}$ is $Y = xP$. This key could itself serve as a credit card number. We also use two cryptographic hash functions $\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$ and $\mathcal{H}_1 : \{0,1\}^* \mapsto \mathbb{G}_1$.

1. The buyer $\mathcal{ID}$ begins by visiting the website of $\mathcal{S}$ and initiating a purchase transaction. The details of the transaction are encapsulated in a request $\mathcal{REQ}$. The tuple $\langle \mathcal{ID}, \mathcal{REQ}, P, Y\rangle$ is sent to $\mathcal{S}$

2. $\mathcal{S}$ generates random $r \leftarrow \mathbb{Z}_q$ and computes $R = rY = rxP$ and $U = r^2P$. It also creates a contract $\mathcal{CON}$ containing the payment amount, transaction date, time and other details (though it will possibly not mention the identity of the seller or the commodity for sale to protect privacy). It sends $\langle \mathcal{CON}, R, U\rangle$ to $\mathcal{ID}$. It is understood that transactions are accepted as valid by the bank only for a short specified deadline (say five minutes) from the time mentioned in the contract.

3. On receiving $\langle \mathcal{CON}, R, U\rangle$, $\mathcal{ID}$ checks that the contract is correct. It then computes $V = \frac{1}{x}U$ and checks that $e(V, V) = e(U, P)$. It this check passes, it computes $Q = \mathcal{H}_1(\mathcal{CON})$, $Z_1 = V + xQ$ and $Z_2 = x\mathcal{H}(Z_1)$. It sends $\langle Z_1, Z_2\rangle$ back to $\mathcal{S}$ and saves $\langle \mathcal{CON}, R\rangle$ in its database until it receives its next credit card statement from the bank. It also keeps a record of $\mathcal{S}$'s reply to the transaction in case a dispute arises.

4. $\mathcal{S}$ computes $Q = \mathcal{H}_1(\mathcal{CON})$ and verifies that $e(Z_1 - rP, P) = e(Q, Y)$ and $e(Z_2, P) = e(\mathcal{H}(Z_1), Y)$. If both checks pass, $\mathcal{S}$ forwards the tuple $\langle Z_1, Z_2, r, Y, \mathcal{ID}, \mathcal{S}, \mathcal{CON}\rangle$ as a payment request to the bank $\mathcal{B}$.

5. On receiving a payment request, $\mathcal{B}$ does the same verification as $\mathcal{S}$; that is, it computes $Q = \mathcal{H}_1(\mathcal{CON})$ and verifies $e(Z_1 - rP, P) = e(Q, Y)$ and $e(Z_2, P) = e(\mathcal{H}(Z_1), Y)$. It also ensures that the $\langle r, \mathcal{ID}\rangle$ pair has not been previously used by checking its database. Finally, the bank checks the date and time specified in $\mathcal{CON}$ and ensures that it is within the specified expiry period (five minutes) of the current time. If all checks pass, $\mathcal{B}$ accepts this transaction, deducts the amount specified from $\mathcal{ID}$'s account, credits that amounts to $\mathcal{S}$'s account, saves the tuple $\langle Z_1, Z_2, \mathcal{ID}, \mathcal{S}, \mathcal{CON}, r\rangle$ in its database and returns `success` to $\mathcal{S}$.

6. The bank's reply is forwarded to the buyer along with a receipt of a successful transaction.

7. If the bank receives another transaction with the same $\langle r, \mathcal{ID} \rangle$ pair in the future, it outputs `failure`. For security reasons, it also saves the corresponding $\langle \mathcal{S}, \mathcal{S}', r, \mathcal{ID} \rangle$ in a *blacklist* where $\mathcal{S}'$ is the identity of the other seller corresponding to the same pair. This blacklist can be used for further investigation if necessary.

8. Sometime in the future, the bank sends $\mathcal{CON}$ in a credit card statement to $\mathcal{ID}$. If some transaction is disputed, $\mathcal{ID}$ reports the corresponding $\langle \mathcal{CON}, R \rangle$ back to the bank, along with some evidence (eg. a transaction receipt with a `failure` response). The bank can easily trace the disputed seller $\mathcal{S}$ using its database after validating that $R = rY$.

7.6 Identity Based Cryptography (IBC)

Using the primitives for identification of section 4.2, any smart card user can dynamically setup a complete Identity Based Cryptosystem (IBC) [15] incorporating both Identity Based Encryption (IBE) and Identity Based Signatures (IBS). The IBE scheme described here is directly taken from [8]. However, to the best of our knowledge, the IBS scheme presented below has not been proposed earlier (we note that the IBS scheme of [16] can also be directly used here).

This smart card user acts as the Key Generating Center (KGC). The interesting feature of our scheme is that the private keys can be distributed over an insecure channel as long as the key-request can be authenticated. The infrastructure is roughly as follows: messages for a user $\mathcal{ID}$ can be encrypted using the public key $\mathcal{ID}$. The private key for decryption is given out by the KGC over an insecure public channel but masked using the hidden-signature scheme of section 7.1. User $\mathcal{ID}$, however, must produce some authenticating information to request a private key. We assume that a signature is used to authenticate user's requests for private keys.

7.6.1 Private Key Distribution

The private key of the user who acts as the KGC for this setup is $x \in \mathbb{Z}_q$ and the corresponding public key is $Y = xP \in \mathbb{G}_1$. Let $\mathcal{H}_1 : \{0,1\}^* \mapsto \mathbb{G}_1$ be a cryptographic hash function. The public key of $\mathcal{ID}_i$ is implicitly understood to be $Q_i = \mathcal{H}_1(\mathcal{ID}_i)$

1. All users must have a prior certified public key to authenticate its requests to the KGC. Each user $\mathcal{ID}_i$ generates $r_i \leftarrow \mathbb{Z}_q$ and computes $R_i = r_i Y = r_i x P$ and $U_i = r_i^2 P$. $\mathcal{ID}_i$ then signs R_i using its certified private key and sends $\langle R_i, U_i \rangle$ to the KGC over an insecure channel.

2. The KGC verifies the signatures and thus authenticates the request of users. For each valid request R_i of $\mathcal{ID}_i$, the KGC computes $V_i = \frac{1}{x} R_i$ and verifies that $e(V_i, V_i) = e(U_i, P)$. If this check passes, it computes $Q_i = \mathcal{H}_1(\mathcal{ID}_i)$ and $Z_i = V_i + xQ_i$ and makes each $\langle Z_i, \mathcal{ID}_i \rangle$ tuple public via an insecure channel.

3. If $\mathcal{ID}_i$ knows corresponding r_i, he/she can compute the private key $xQ_i = Z_i - r_i P$ after authenticating it by checking that $e(xQ_i, P) = e(Q_i, Y)$. The encryption/decryption can be done exactly as described in [8] after this step (described next). The zero-knowledge property ensures that only the right user can compute the private key from Z_i.

7.6.2 Identity Based Encryption (IBE)

We briefly describe the identity based encryption scheme here (further details can be obtained from [8]). Let $\mathcal{H}_2 : \mathbb{G}_2 \mapsto \{0,1\}^k$ be a cryptographic hash function. A random k bit message M for $\mathcal{ID}_i$ is encrypted as follows:

1. ***Encryption***: Generate $\alpha \leftarrow \mathbb{Z}_q$, compute $Q_i = \mathcal{H}_1(\mathcal{ID}_i)$, $C_1 = M \oplus \mathcal{H}_2(e(\alpha Q_i, Y))$ and $C_2 = \alpha P$. The ciphertext (C_1, C_2) is sent to $\mathcal{ID}_i$.

2. ***Decryption***: $\mathcal{ID}_i$ decrypts $M = C_1 \oplus \mathcal{H}_2(e(C_2, xQ_i))$.

7.6.3 Identity Based Signatures (IBS)

In this section, we propose a novel IBS scheme using bilinear pairings. The setup of the scheme is exactly as the one for the above IBE scheme. We will use the same hash function $\mathcal{H}_1 : \{0,1\}^* \mapsto \mathbb{G}_1$ in this scheme. As before the public key of $\mathcal{ID}_i$ is $\langle Q_i, Y, P\rangle$ and the private key is xQ_i where $Q_i = \mathcal{H}_1(\mathcal{ID}_i)$.

1. ***Signing***: To sign a message $M \in \{0,1\}^*$, user $\mathcal{ID}_i$ generates $\alpha \leftarrow \mathbb{Z}_q$, computes $S = xQ_i + \alpha\mathcal{H}_1(M)$ and $T = \alpha P$. The tuple $\langle M, (S, T)\rangle$ is a valid message-signature pair.

2. ***Verification***: To verify a signature, we check that $e(S, P) = e(Q_i, Y) \cdot e(\mathcal{H}_1(M), T)$. To the best of our knowledge, this IBS variant has not been proposed before.[6]

The scheme is secure in the random oracle model under the Diffie-Hellman assumption in $\mathbb{G}_1$. The security of this scheme will be further detailed in the full version of this paper.

8 Summary

Scheme	**Ref. (Sec.)**	**Shared Params**	**Hash Functions**	**Processing**		**Message Size** (bits)	
				Initiator	Responder	First	Second
Single User Identification	4.2	none	none	3 exp 2 pairing	2 exp 2 pairing	344	344
Multi-User Identification	6	$P, \mathbb{G}_1, \mathbb{G}_2$	none	$2n+1$ exp 2 pairing	2 exp 2 pairing	$344n$	$171(n+1)$*
Hidden Signatures	7.1	none	$\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$ 2 exp	3 exp 2 pairing	$2^{\#}$ 2 pairing	344	344
Plaintext-aware Encryption	7.2	none	$\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$	5 exp 4 pairing	1 exp	859	
Plaintext-aware Signcryption	7.2	$\mathbb{G}_1, \mathbb{G}_2$	$\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$	4 exp 4 pairing	1 exp	688	
Deniable signcryption	7.3	$\mathbb{G}_1, \mathbb{G}_2$	none	3 exp 2 pairing	3 exp 2 pairing	344	513
2-round key agreement	7.4	$\mathbb{G}_1, \mathbb{G}_2$	$\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$	5 exp 4 pairing	4 exp 4 pairing	513	513
Credit-card payment	7.5	none	$\mathcal{H} : \mathbb{G}_1 \mapsto \mathbb{G}_1$ $\mathcal{H}_1 : \{0,1\}^* \mapsto \mathbb{G}_1$	4 exp 2 pairing	4 exp 4 pairing	$344^{\#}$	344
IBS	7.6.3	none	$\mathcal{H}_1 : \{0,1\}^* \mapsto \mathbb{G}_1$	2 exp	3 pairing	344	

\# Here the initiator is $\mathcal{S}$.

* For simplicity, we do not count the communication overhead incurred in computing the joint proof in the multi-user identification. Here n is the number of users.

Assuming $q \approx 2^{171}$, each pairing computation takes ≈ 8.5 ms and each exponentiation takes ≈ 1.5 ms on the hardware mentioned.

Table 1: Summary

In this paper, we proposed the notion of *zero knowledge blind identification*. Informally, in such a protocol, an honest prover reveals only one (intended) bit of information to an honest verifier and reveals less than that information to a dishonest verifier. In effect, using our scheme, any user can correctly identify to a random server and a passive adversary cannot learn anything about the outcome of the

[6] A previously proposed IBS scheme [16] uses a hash function $\mathcal{H}_3 : \{0,1\}^* \mapsto \mathbb{Z}_q$ and works as follows: To sign a message $M \in \{0,1\}^*$, $\mathcal{ID}_i$ generates $\alpha \leftarrow \mathbb{Z}_q$ and computes $S = (\mathcal{H}_3(M) + \alpha)xQ_i$ and $T = \alpha P$. The signature is $\langle S, T\rangle$ which is verified by checking that $e(S, P) = e(\mathcal{H}_3(M)Q_i + T, Y)$. Our scheme has the advantage of requiring only one hash function (at the cost of one extra pairing computation).

identification. Hence we coin the term blind identification. To the best of our knowledge, this blinding property is unique to our scheme.

Referring to the definitions of sections 4 and 4.2, essentially, the security of our protocol relies on the hardness of the LDDHP (i.e. deciding if $Z = (r + xy)P$ for given P, xP, rxP, yP, Z). Although, this is not a well studied hard problem like the DHP, we feel reasonably confident that it is computationally intractable.

In section 7, we show how these simple identification primitives can be used for constructing complex mechanisms like key agreement, digital signatures, encryption and signcryption. As a simple application of our smart card scheme, we propose a model for on line credit card and cheque transactions. The protocol can be used in conjunction with the Secure Electronic Transaction (SET) specification or in a completely different infrastructure. As some other applications, we mention subliminal identification, designated verifier proofs and multiuser authentication. For optimal security, the primitives for signing are best implemented in a tamper proof chip supporting elliptic curve point addition and doubling operations. As observed, all the verification primitives require one or two pairing computations and deal with public keys only. Consequently, they are not restricted to a secure tamper proof device. We refer the reader to [7] for details on constructing the hash functions used here.

Finally, we summarize the efficiency and viability of our scheme. For this discussion, we will assume that $q \approx 2^{171}$ so that the best known algorithms for computing discrete logarithms in $\mathbb{G}_1$ require $\approx 2^{85}$ operations and are therefore impractical. The benchmarks of [17] indicate that each pairing operation using these parameters takes ≈ 8.6 ms and each elliptic curve point exponentiation takes ≈ 1.5 ms. These results were obtained on a desktop PC with an AMD Athlon 2100+ 1.8 GHz, 1 GB RAM and an IBM 7200 RPM, 40 GB, Ultra ATA/100 hard drive [17]. Using these values we obtain the following results. Table 1 summarizes our results.

References

[1] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[2] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):690–728, 1991.

[3] Mihir Bellare and Oded Goldreich. On defining proofs of knowledge. *Lecture Notes in Computer Science*, 740:390–420, 1993.

[4] M. Bellare, S. Micali, and R. Ostrovsky. The (true) complexity of statistical zero knowledge. In *STOC '90: Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 494–502, New York, NY, USA, 1990. ACM Press.

[5] Amitabh Saxena and Ben Soh. Bounded prover zero-knowledge in two-rounds, 2005. Unpublished Manuscript. url: http://homepage.cs.latrobe.edu.au/asaxena/saxena05bounded.pdf.

[6] Amos Fiat and Adi Shamir. How to prove yourself: practical solutions to identification and signature problems. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 186–194, London, UK, 1987. Springer-Verlag.

[7] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT '01: Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532, London, UK, 2001. Springer-Verlag.

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229. Springer-Verlag, 2001.

[9] Paulo S. L. M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *CRYPTO '02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 354–368, London, UK, 2002. Springer-Verlag.

[10] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.

[11] Jean-Sébastien Coron and David Naccache. Boneh et al.'s k-element aggregate extraction assumption is equivalent to the diffie-hellman assumption. In Chi-Sung Laih, editor, *ASIACRYPT*, volume 2894 of *Lecture Notes in Computer Science*, pages 392–397. Springer, 2003.

[12] Moses Liskov, Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Adam Smith. Mutually independent commitments. *Lecture Notes in Computer Science*, 2248:385+, 2001.

[13] Jonathan Herzog, Moses Liskov, and Silvio Micali. Plaintext awareness via key registration. Technical report, MIT Laboratory for Computer Science, February 2003.

[14] Hugo Krawczyk. Hmqv: A high-performance secure diffie-hellman protocol. Cryptology ePrint Archive, Report 2005/176, 2005.

[15] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[16] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap diffie-hellman groups. In Yvo Desmedt, editor, *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 18–30. Springer, 2003.

[17] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. Cryptology ePrint Archive, Report 2005/028, 2005.